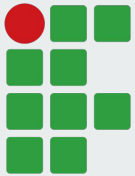
SINGLETON



INSTITUTO FEDERAL
Paraná

Disciplina: Análise e Projeto de Algoritmos
Alunos: Renã Borba
Vinicius Mordhorst

Oque é?

O Singleton é um padrão de projeto criacional que permite a você garantir que uma classe tenha apenas uma instância, enquanto provê um ponto de acesso global para essa instância.

Log

Logs normalmente são utilizados por quase todas as classes de um sistema, e não retornam nenhuma informação que afeta o comportamento da aplicação. Este é um caso no qual o singleton pode ser bem empregado.

Problema/Por que usar?

Contexto: Suponha uma classe `Logger`, usada para registrar as operações realizadas em um sistema. Um uso dessa classe é mostrado a seguir:

```
 void f() {
   Logger log = new Logger();
   log.println("Executando f");
   ...
}
void g() {
   Logger log = new Logger();
   log.println("Executando g");
   ...
}
void h() {
   Logger log = new Logger();
   log.println("Executando h");
   ...
}
```

Solução

Todas as implementações do Singleton tem esses dois passos em comum:

- Fazer o construtor padrão privado, para prevenir que outros objetos usem o operador `new` com a classe singleton.
- Criar um método estático de criação que age como um construtor. Esse método chama o construtor privado por debaixo dos panos para criar um objeto e o salva em um campo estático. Todas as chamadas seguintes para esse método retorna o objeto em cache.

Solução

Solução: A solução para esse problema consiste em transformar a classe `LOGGER` em um **Singleton**. Esse padrão de projeto define como implementar classes que terão, como o próprio nome indica, no máximo uma instância. Mostramos a seguir a versão de `LOGGER` que funciona como um Singleton:

```
class Logger {

  private Logger() {} // proíbe clientes chamar new Logger()

  private static Logger instance; // instância única

  public static Logger getInstance() {
    if (instance == null) // 1a vez que chama-se getInstance
      instance = new Logger();
    return instance;
  }

  public void println(String msg) {
    // registra msg no console, mas poderia ser em arquivo
    System.out.println(msg);
  }
}
```

Prós e contras

- ✓ Você pode ter certeza que uma classe só terá uma única instância.
- ✓ Você ganha um ponto de acesso global para aquela instância.
- ✓ O objeto singleton é inicializado somente quando for pedido pela primeira vez.

- ✗ Viola o *princípio de responsabilidade única*. O padrão resolve dois problemas de uma só vez.
- ✗ O padrão Singleton pode mascarar um design ruim, por exemplo, quando os componentes do programa sabem muito sobre cada um.
- ✗ O padrão requer tratamento especial em um ambiente multithreaded para que múltiplas threads não possam criar um objeto singleton várias vezes.
- ✗ Pode ser difícil realizar testes unitários do código cliente do Singleton porque muitos frameworks de teste dependem de herança quando produzem objetos simulados. Já que o construtor da classe singleton é privado e sobrescrever métodos estáticos é impossível na maioria das linguagem, você terá que pensar em uma maneira criativa de simular o singleton. Ou apenas não escreva os testes. Ou não use o padrão Singleton.

Referências

- Marco Tulio Valente. (n.d.). Engenharia de software moderna. Recuperado de https://engsoftmoderna.info/cap6.html
- Refactoring Guru. (n.d.). Padrão de projeto Singleton. Refactoring Guru. Recuperado de https://refactoring.guru/pt-br/design-patterns/singleton

  Exemplo: `https://dontpad.com/singletonvkm`